Theoretical Computer Science 410 (2009) 4856–4863

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Scheduling with families of jobs and delivery coordination under job availability☆

Shisheng Li, Jinjiang Yuan *

Department of Mathematics, Zhengzhou University, Zhengzhou, Henan 450052, People's Republic of China

ARTICLE INFO

Article history:
Received 26 February 2009
Received in revised form 10 June 2009
Accepted 13 June 2009
Communicated by D.-Z. Du

Keywords:
Scheduling
Families of jobs
Batching
Delivery

ABSTRACT

We consider in this paper the scheduling of families of jobs in which both processing and delivery are coordinated together. Only one vehicle is available to deliver the jobs to specified customers. The jobs can be processed together to form processing batches on the machine and setups of batches are required when the machine is changing from one family to another. Jobs from different families cannot be transported together by the vehicle. The objective is to minimize the time when the vehicle finishes delivering the last delivery batch to its customer and returns to the machine. We propose an $O(n \log n)$-time optimal algorithm for the scheduling problem under the group technology assumption. For the scheduling problem without the group technology assumption, we show that the problem is NP-hard and give an $O(f^2 n^f)$-time dynamic programming algorithm, where n is the number of jobs, and f is the number of families; we also provide a heuristic algorithm with a performance ratio of $3/2$.

1. Introduction

We are given a set of n nonpreemptive jobs $J_1, J_2, \ldots, J_n$ that are classified into f families $F_1, F_2, \ldots, F_f$. All jobs are available at time 0. The jobs are processed on a single machine. Specially, a set of jobs in the same family can be processed together to form processing batches on the machine. Setups of batches are required when the machine is changing from a processing batch of one family to a processing batch of another family. There is a capacitated vehicle to transport the processing completed jobs to specified customers. We assume that jobs of the same family have identical amount of physical space in a transportation vehicle and belong to a specified customer. We further assume that, jobs from distinct families cannot be transported together by the vehicle in a delivery batch. The goal is to find a schedule to minimize the time when the vehicle finishes delivering the last delivery batch to its customer and returns to the machine.

There are two variants of scheduling the families of jobs depending on when the jobs become available for delivery. Under the batch availability, a job becomes available only when the batch to which it belongs has been processed. An alternative assumption is the job availability (usually known in the literature as item availability), in which a job becomes available immediately after its processing is completed. In this paper, we adopt the assumption of job availability.

We follow the three-field $\alpha|\beta|\gamma$ notation of Chang and Lee [5] to denote the problem under study, with extensions to include families of jobs and batch deliveries. Then the problem is denoted by $1 \to D, f|v = 1, s_i, c_i, t_i|C_{\max}$. In the α field, $1 \to D, f$ means that the jobs are processed on a single machine and processing completed jobs are delivered to customers located in f areas. In the β field, $v = 1$ means that there is only one vehicle to deliver the jobs, and s_i, c_i, and t_i denote the

☆ Research supported by NSFC (10671183), NFSC-RGC (70731160633) and SRFDP (20070459002).
* Corresponding author. Tel.: +86 371 67767835.
E-mail address: yuanjj@zzu.edu.cn (J. Yuan).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.06.025

setup time, the capacity of the vehicle and the delivery time for jobs in family F_i $(1 \leq i \leq f)$, respectively. In the γ field, $C_{\max}$ denotes the time when the vehicle finishes delivering the last delivery batch to its customer and returns to the machine.

Hall and Potts [9] considered a variety of scheduling, batching, and delivery scenarios that arise in an arborescent supply chain. The objective is to minimize the overall processing and delivery cost. Transportation capacity was not considered in their models. They showed that if decision makers at different stages of a supply chain make poorly coordinated decisions at the operational level, substantial inefficiencies may result. Erengüc et al. [7] emphasized the operational aspects of supply chains and the need for decision making at the supplier, plant, and distribution stages of a supply chain. Lee and Chen [12] studied the problem combining machine scheduling and finished job delivery together. Jobs are delivered in batches by vehicles. Both transportation times and vehicle capacity were considered in their models in which all jobs take the same amount of physical space in a transportation vehicle. Chang and Lee [5] extended Lee and Chen's work to the situation where each finished job has a different size; hence, assigning jobs to delivery batches amounts to solving the bin-packing problem. They proposed a heuristic with a worst-case analysis for each special scenario associated with different processing settings and customers. Li et al. [13] developed a single machine scheduling model that incorporates routing decisions of a delivery vehicle that serves customers at different locations. The objective is to minimize the sum of job arrival times. They proposed dynamic programming algorithms for the special case with a single customer and for the general case with arbitrary customers. To learn more about research results on this aspect, the reader is referred to see Ahmadi et al. [1], Thomas and Griffin [19], Mazdeh et al. [15], Zhong et al. [21], and Chen and Lee [6].

Another line of research related to the problem under research focuses on scheduling with families of jobs. Bruno and Downey [4] studied single machine problems with deadlines and setup times. They showed that the problem is strongly NP-hard. Baker and Magaine [3] examined a single machine scheduling with sequence-independent setups between different families of jobs. For this NP-hard problem of minimizing maximum lateness with job families, they exploited special structure to compress the effective problem size by creating composite jobs and accelerated the enumeration with dominance properties and lower bounds. Jin et al. [11] extended Baker and Magaine's work to the situation where sequence-dependent setups were required between different families of jobs. They proposed a simulated annealing algorithm with the new neighborhood to solve the problem. Liaee and Emmons [14] reviewed scheduling theory concerning the processing of several families of jobs on single or parallel facilities. For various performance measures, they classified the different problems as NP-hard, efficiently solvable or open. Schaller [18] considered a single machine scheduling problem to minimize total tardiness when family setups exist, and proposed optimal branch-and-bound procedures.

There exist many research results on scheduling and batching problems (see the survey papers by Potts and Wassenhove [17], Webster and Baker [20], Potts and Kovalyov [16], and Allahverdi et al. [2]). However, these researches did not take transportation times into consideration, i.e., they assumed that delivery of a job can be made whence its processing is completed.

The remainder of this paper is organized as follows. In Section 2, we introduce some notations and three useful lemmas. In Section 3, we study the scheduling problem under the group technology (GT) assumption and propose an $O(n \log n)$-time optimal algorithm for $1 \to D, f|v = 1, s_i, c_i, t_i, GT|C_{\max}$. Section 4 discusses the scheduling problem without the GT assumption. In Section 4.1, we show that problem $1 \to D, f|v = 1, s_i, c_i, t_i|C_{\max}$ is NP-hard. In Section 4.2, we give an $O(f^2 n^f)$-time dynamic programming algorithm, where n is the number of jobs, and f is the number of families. In Section 4.3, we provide a heuristic algorithm with a performance ratio of 3/2.

2. Preliminaries

In this section, we first give some notations that will be used in this paper and then give three lemmas for the characterization of the delivery batch structure.

For each $i = 1, 2, \ldots, f$, the following notations will be used:

- n_i the number of jobs in family F_i
- s_i the setup time for family F_i
- t_i the delivery time of jobs in F_i, i.e., t_i is the time needed for a round of transportation of the vehicle to deliver the jobs in F_i to their specified customer and return to the machine
- c_i the capacity of the vehicle for family F_i, i.e., at most c_i jobs in F_i can be delivered in a round of transportation
- J_{ij} jth job in family $F_i, j = 1, 2, \ldots, n_i$
- p_{ij} the processing time of job $J_{ij}, j = 1, 2, \ldots, n_i$
- P_i the total processing times of jobs in family F_i, i.e., $P_i = \sum_{1 \leq j \leq n_i} p_{ij}$

Note that the objective value $C_{\max}$ is a regular measure of performance. So we can assume that all jobs are processed on the machine without idle time. The following lemma can be easily proved by the pairwise interchange argument.

Lemma 2.1. *For $1 \to D, f|v = 1, s_i, c_i, t_i|C_{\max}$, there exists an optimal schedule such that the jobs within each family are processed and delivered in nondecreasing order of their processing times (SPT).* □

We called a schedule π an SPT schedule if, in schedule π, the jobs within each family are processed and delivered in SPT order. From Lemma 2.1, we can see that an optimal SPT schedule must be an optimal schedule.

For simplicity, we re-index the jobs within each family according to the SPT rule, i.e., $p_{i1} \leq p_{i2} \leq \cdots, \leq p_{in_i}$ for $1 \leq i \leq f$. Furthermore, for each i with $1 \leq i \leq f$, we define q_i and u_i to be the integers with $n_i = c_i q_i + u_i$ and $0 < u_i \leq c_i$.

Lemma 2.2. *For $1 \to D, f|v = 1, s_i, c_i, t_i|C_{\max}$, there exists an optimal SPT schedule satisfying the following properties.*

(i) *Jobs assigned to one delivery batch are processed consecutively in a processing batch on the machine.*
(ii) *Early processed jobs are delivered no later than those processed later.*
(iii) *The delivery batch of each family can be determined by the first-only-empty (FOE) [10] batch rule. Specially, for each family F_i, $1 \leq i \leq f$, there are $q_i + 1$ delivery batches; the first delivery batch contains u_i jobs, and each of the other delivery batches contains exactly c_i jobs.*

Proof. (i) and (ii) can be proved by the pairwise interchange argument. To prove (iii), let π be an optimal SPT schedule satisfying (i) and (ii). If a delivery batch of some family (say F_j), with exception of the first delivery batch, contains less than c_j jobs, we can always fill the delivery batch with more jobs from the earlier delivery batches of the same family without increasing the objective value. Repeating this procedure family by family at most f times, we can obtain an optimal schedule satisfying (iii). □

Lemma 2.3. *Problem $1 \to D, f|v = 1, s_i, c_i, t_i|C_{\max}$ can be reduced to problem $1 \to D, f|v = 1, s_i, c_i = 1, t_i|C_{\max}$ in linear time.*

Proof. A direct consequence of Lemma 2.2. □

3. Scheduling under GT assumption

In this section, we consider problem $1 \to D, f|v = 1, s_i, c_i, t_i|C_{\max}$ with the group technology (GT) assumption [14]. Under the GT assumption, the jobs in a family must be processed consecutively. Thus, the problem may be separated into finding an optimal processing order of jobs in each family, and finding the optimal sequence of families. For ease of exposition, we denote the problem under GT assumption as $1 \to D, f|v = 1, s_i, c_i, t_i, GT|C_{\max}$. We refer a feasible schedule subject to the GT assumption as a GT schedule.

It can be observed that Lemmas 2.1–2.3 are still valid for problem $1 \to D, f|v = 1, s_i, c_i, t_i, GT|C_{\max}$.

Lemma 3.1. *For $1 \to D, f|v = 1, s_i, c_i, t_i, GT|C_{\max}$, there is an optimal GT schedule (called SPT-NO-IDLE schedule) such that the jobs within each family are processed according to SPT rule on the machine, and the delivery batches of each family are consecutively transported without idle time.*

Proof. The first statement follows from Lemma 2.1. To prove the second statement, let π be an optimal SPT schedule under GT assumption. From Lemma 2.2(ii), we can assume that, for each family F_j, there are no delivery batches of other families between the transportation of the first delivery batch and the last delivery batch of F_j. So we can always delay the departure times of some delivery batches of F_j such that the vehicle is always busy from the departure of the first delivery batch of family F_j until the delivery completion time of F_j. The result holds. □

For each family F_i, $1 \leq i \leq f$, we use π_i to denote an optimal SPT-NO-IDLE schedule of the problem restricted on family F_i. The corresponding optimal objective value of π_i is denoted by $C_{\max}(\pi_i)$. Recall that $q_i + 1$ is the number of delivery batches in some optimal SPT-NO-IDLE schedule. We define an associated 3-tuple (a_i, b_i, λ_i) for each family F_i by the following way:

- $a_i = s_i + P_i$.
- $b_i = (q_i + 1)t_i$.
- $\lambda_i = b_i - (C_{\max}(\pi_i) - a_i) = s_i + P_i + (q_i + 1)t_i - C_{\max}(\pi_i)$.

Under the above definition, we can see that a_i is the sum of the processing times of jobs and setup time of F_i, b_i is the sum of the delivery times of the delivery batches of F_i, and λ_i is the maximal overlap value of the processing time and the delivery time for family F_i. It can observed that

$$C_{\max}(\pi_i) = a_i - \lambda_i + b_i \quad \text{for } 1 \leq i \leq f. \tag{1}$$

The above discussion also tells us that, to guarantee an optimal SPT-NO-IDLE schedule of problem $1 \to D, f|v = 1, s_i, c_i, t_i, GT|C_{\max}$, if the setup of family F_i starts at time h_i, then the departure time of the first delivery batch of F_i can be given by $\max\{h_i + a_i - \lambda_i, \delta_i\}$, where δ_i is the delivery completion time of the family directly before F_i. If F_i is the first family in the schedule, δ_i is defined to be 0.

Algorithm GT

Step 1. For each family F_i, $1 \leq i \leq f$, re-index the jobs according to the SPT rule and use the FOE batch rule to determine the delivery batches. Then evaluate $C_{\max}(\pi_i)$.
Step 2. Associate a 3-tuple (a_i, b_i, λ_i) for each family F_i as defined above.
Step 3. Schedule each family as a single processing batch in the following way:
Step 3.1 Partition the families into two subsets A_1 and A_2 by setting $A_1 = \{F_i : a_i \leq b_i\}$ and $A_2 = \{F_i : a_i > b_i\}$.
Step 3.2 Process first the families in A_1 in nondecreasing order of $a_i - \lambda_i$, then the families in A_2 in nonincreasing order of $b_i - \lambda_i$ on the machine. Suppose that the setup of a family F_j starts at time h_j, $1 \leq j \leq f$. At the first time t with $t \geq h_j + a_j - \lambda_j$ and the vehicle being available, start to transport the delivery batches of F_j consecutively without idle time.

It can be observed that the running time of algorithm GT is $O(n \log n)$.

Theorem 3.2. *Algorithm GT is optimal for* $1 \to D, f|v = 1, s_i, c_i, t_i, GT|C_{\max}$.

Proof. Let π be an optimal GT schedule satisfying Lemma 3.1 but does not follow algorithm GT, then there exists at least two families F_j and F_k such that family F_k follows immediately after family F_j, and one of following three cases occurs: (a) F_j belongs to A_2, and F_k belongs to A_1; (b) Both F_j and F_k belong to A_1, and $a_j - \lambda_j > a_k - \lambda_k$; (c) Both F_j and F_k belong to A_2, and $b_j - \lambda_j < b_k - \lambda_k$.

In the following, we will prove that interchanging the order of family F_j and F_k in π does not increase the objective value, so the result holds by repeatedly interchanging families that do not follow algorithm GT.

Suppose that family F_l (if any) precedes immediately before F_j and family F_s follows immediately after F_k in π. That is, F_l, F_j, F_k and F_s are four consecutively processed families. In the case that F_l or F_s does not exist, it is assumed to be a dummy family with setup time, processing time and delivery time being 0. We perform an adjacent pairwise interchange of family F_j and F_k, leaving the remaining families in their original positions. The resulted new schedule is denoted by π'.

Under π (π'), we use K_{l1} (K'_{l1}), K_{j1} (K'_{j1}), K_{k1} (K'_{k1}) and K_{s1} (K'_{s1}) to denote the processing completion time of F_l, F_j, F_k and F_s, respectively, and use K_{l2} (K'_{l2}), K_{j2} (K'_{j2}), K_{k2} (K'_{k2}) and K_{s2} (K'_{s2}) to denote the delivery completion time of F_l, F_j, F_k and F_s, respectively.

Obviously, we have $K_{l1} = K'_{l1}$ and $K_{l2} = K'_{l2}$. After exchanging the positions of families F_j and F_k, the starting time of family F_s on the machine is not affected, i.e., under π and π', the starting time of family F_s on the machine is $K_{l1} + a_j + a_k$. In the following, we will prove that after exchange the positions of families F_j and F_k, the departure time of the first delivery batch of family F_s will not delay.

Under π, the departure time of the first delivery batch of family F_s can be expressed as $\max\{K_{k1} + a_s - \lambda_s, K_{k2}\} = \max\{K_{l1} + a_j + a_k + a_s - \lambda_s, K_{k2}\}$. Under π', the departure time of the first delivery batch of family F_s can be expressed as $\max\{K'_{j1} + a_s - \lambda_s, K'_{j2}\} = \max\{K'_{l1} + a_k + a_j + a_s - \lambda_s, K'_{j2}\}$. Note that $K_{l1} = K'_{l1}$, we only need to show that $K'_{j2} \le K_{k2}$.

From the execution of algorithm GT, we have

$$\begin{aligned} K_{k2} &= \max\{K_{j2}, K_{j1} + a_k - \lambda_k\} + b_k \\ &= \max\{\max\{K_{l2}, K_{l1} + a_j - \lambda_j\} + b_j, K_{l1} + a_j + a_k - \lambda_k\} + b_k \\ &= \max\{K_{l2} + b_j + b_k, K_{l1} + a_j - \lambda_j + b_j + b_k, K_{l1} + a_j + a_k - \lambda_k + b_k\}. \end{aligned} \tag{2}$$

Similarly,

$$\begin{aligned} K'_{j2} &= \max\{K'_{k2}, K'_{k1} + a_j - \lambda_j\} + b_j \\ &= \max\{\max\{K'_{l2}, K'_{l1} + a_k - \lambda_k\} + b_k, K'_{l1} + a_k + a_j - \lambda_j\} + b_j \\ &= \max\{K'_{l2} + b_k + b_j, K'_{l1} + a_k - \lambda_k + b_k + b_j, K'_{l1} + a_k + a_j - \lambda_j + b_j\}. \end{aligned} \tag{3}$$

Since $K_{l2} = K'_{l2}$, the first parts of the last "max" in (2) and (3) are equal. So, we only need to show that

$$\max\{K'_{l1} + a_k - \lambda_k + b_k + b_j, K'_{l1} + a_k + a_j - \lambda_j + b_j\} \le \max\{K_{l1} + a_j - \lambda_j + b_j + b_k, K_{l1} + a_j + a_k - \lambda_k + b_k\}. \tag{4}$$

Recall that $K_{l1} = K'_{l1}$. Subtracting $K_{l1} + a_j + a_k + b_j + b_k - \lambda_j - \lambda_k$ from both sides of (4), we have an equivalent inequality $\max\{-a_j + \lambda_j, -b_k + \lambda_k\} \le \max\{-a_k + \lambda_k, -b_j + \lambda_j\}$, or equivalently,

$$\min\{a_k - \lambda_k, b_j - \lambda_j\} \le \min\{a_j - \lambda_j, b_k - \lambda_k\}. \tag{5}$$

In case (a), according to algorithm GT, we have $a_j > b_j$ and $a_k \le b_k$. Then $a_k - \lambda_k \le b_k - \lambda_k$ and $b_j - \lambda_j \le a_j - \lambda_j$, and so, (5) holds.

In case (b), according to algorithm GT, we have $a_j \le b_j$ and $a_k \le b_k$. Recall that $a_j - \lambda_j > a_k - \lambda_k$. Then $\min\{a_k - \lambda_k, b_j - \lambda_j\} \le a_k - \lambda_k \le \min\{a_j - \lambda_j, b_k - \lambda_k\}$, and so, (5) holds.

In case (c), according to algorithm GT, we have $a_j > b_j$ and $a_k > b_k$. Recall that $b_j - \lambda_j < b_k - \lambda_k$. Then $\min\{a_k - \lambda_k, b_j - \lambda_j\} \le b_j - \lambda_j \le \min\{a_j - \lambda_j, b_k - \lambda_k\}$, and so, (5) holds. The result follows. □

Remark 3.3. For problem $1 \to D, f|v = 1, s_i, c_i, t_i|C_{\max}$ without GT assumption, whence the processing batches of an optimal schedule are given, then the optimal schedule can be found by using algorithm GT.

4. Scheduling without GT assumption

In this section, we study the scheduling problem without the GT assumption. By Lemma 2.3, we can reduce problem $1 \to D, f|v = 1, s_i, c_i, t_i|C_{\max}$ to $1 \to D, f|v = 1, s_i, c_i = 1, t_i|C_{\max}$ in linear time.

For problem $1 \to D, f|v = 1, s_i, c_i = 1, t_i|C_{\max}$, we first show its NP-hardness, then we give an $O(f^2 n^f)$-time dynamic programming algorithm and provide a heuristic algorithm with a performance ratio of 3/2.

4.1. NP-hardness proof

We need the following NP-complete Equal-Size Partition problem (see Garey and Johnson [8]).

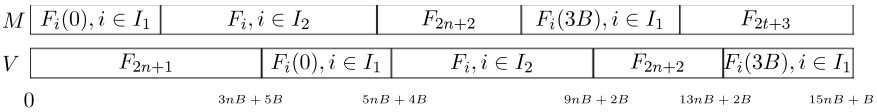

Fig. 1. Configuration of the optimal schedule in Theorem 4.1.

Equal-Size Partition: Given a set of $2n + 1$ positive integers $a_1, a_2, \ldots, a_{2n}$ and B such that $\sum_{i=1}^{2n} a_i = 2B$, does there exist a partition I_1 and I_2 of the index set of $S = \{1, \ldots, 2n\}$ such that $|I_j| = n$ and $\sum_{i \in I_j} a_i = B$ for $j = 1, 2$?

Theorem 4.1. *Problem* $1 \to D, f|v = 1, s_i, c_i = 1, t_i|C_{\max}$ *is NP-hard.*

Proof. The decision version of the scheduling problem is clearly in NP. To prove the NP-hardness, we use the NP-complete Equal-Size Partition for the reduction.

Given an arbitrary instance $(a_1, \ldots, a_{2n}; B)$ of Equal-Size Partition, we construct an instance of decision version of the scheduling problem as follows.

- There are $f = 2n + 3$ families of jobs $F_1, \ldots, F_{2n+3}$.
- Each of the first $2n$ families F_i $(1 \le i \le 2n)$ consists two jobs J_{i1} and J_{i2}. These families are called normal families. For each i with $1 \le i \le 2n$, we have $s_i = B + a_i$, $t_i = 2B - a_i$, $p_{i1} = 0$ and $p_{i2} = 3B$.
- Family F_{2n+1}, called head-family, contains only one job J_{2n+1}. We have $s_{2n+1} = 0$, $p_{2n+1} = 0$ and $t_{2n+1} = 3nB + 5B$. The head-family F_{2n+1} guarantees that the first delivery batch starts its delivery at time 0.
- Family F_{2n+2}, called partition-family, contains only one job J_{2n+2}. We have $s_{2n+2} = 0$, $p_{2n+2} = 4nB$ and $t_{2n+2} = 4nB$. The partition-family F_{2n+2} is used to partition the normal families into two parts according to their schedule.
- Family F_{2n+3}, called tail-family, also contains only one job J_{2n+3}. We have $s_{2n+3} = 0$, $p_{2n+3} = 2nB - 2B$ and $t_{2n+3} = 0$. The tail-family will guarantee that the total setup time and processing time is equal to the total delivery time.
- The threshold value is given by $Y = 15nB + B$ which is the total delivery time of the delivery batches.
- The decision asks whether there is a schedule π such that $C_{\max}(\pi) \le Y$.

Clearly, the construction can be done in polynomial time. We show in the sequel that the instance of Equal-Size Partition has a solution if and only if there is schedule π for the scheduling instance such that $C_{\max}(\pi) \le Y$.

Assume that (I_1, I_2) is a solution of the instance of Equal-Size Partition. That is, (I_1, I_2) is a partition of $S = \{1, \ldots, 2n\}$ with $|I_1| = |I_2| = n$ and $\sum_{i \in I_1} a_i = \sum_{i \in I_2} a_i = B$. Then we can define a schedule π by the following way.

Each family F_i with $i \in I_1$ is partitioned into two processing batches $F_i(0)$ and $F_i(3B)$, where $F_i(0)$ consists the job J_{i1} with processing time 0, and $F_i(3B)$ consists the job J_{i2} with processing time $3B$. Each of the other families acts as a single processing batch. The processing batches are processed by the order

$$F_{2n+1} \to F_i(0), i \in I_1 \to F_i, i \in I_2 \to F_{2n+2} \to F_i(3B), i \in I_1 \to F_{3n+3}.$$

Since $c_i = 1$ for each family, each job acts as a delivery batch. The delivery batches are transported in the same order of the processing of the jobs. We use M to denote the machine and V to denote the vehicle. Then schedule π can be indicated by Fig. 1 with the zero-time processing of F_{2n+1} being omitted.

It can be verified that schedule π has objective value $C_{\max}(\pi) = Y$.

Conversely, assume that the scheduling instance has an optimal schedule π with $C_{\max}(\pi) \le Y$. Note that the total delivery time is $\sum_{i=1}^{3} t_{2n+i} + \sum_{i=1}^{2t} 2t_i = 15nB + B = Y$. Then we have $C_{\max}(\pi) \ge Y$. Consequently, $C_{\max}(\pi) = Y$ and the vehicle is always busy from time 0 to time Y. We define a partition (I_1, I_2) of $S = \{1, \ldots, 2n\}$ by setting

$$I_1 = \{i : 1 \le i \le 2n,\ F_i \text{ is processed as two batches in } \pi\},$$

$$I_2 = \{i : 1 \le i \le 2n,\ F_i \text{ is processed as a single batch in } \pi\}.$$

As above, we suppose that F_i with $i \in I_1$ is partitioned into two processing batches $F_i(0)$ and $F_i(3B)$, where $F_i(0)$ consists the job J_{i1} with processing time 0, and $F_i(3B)$ consists the job J_{i2} with processing time $3B$. Note that the processing batch structure has been determined, by Remark 3.3 and the simple arguments, we can assume that π satisfies the following conditions:

(a) Family F_{2n+1}, with $s_{2n+1} = p_{2n+1} = 0$, is processed first starting at time 0, since the first delivery batch must be delivered at time 0.
(b) Family F_{2n+3} is processed last. Otherwise we can interchange it with the jobs from other families without increase the objective value, since $t_{2n+3} = 0$.
(c) The jobs in F_i, $1 \le i \le 2n$, and F_{2n+2} are processed in the following order (by Remark 3.3 and algorithm GT):

$$F_i(0), i \in I_1 \to F_i, i \in I_2 \to F_{2n+2} \to F_i(3B), i \in I_1.$$

(d) The vehicle transports jobs in the interval $[0, 15nB + B]$ without idle time.
(e) The sum of processing times of jobs and the setup time of processing batches on the machine cannot exceed $15nB + B = Y$.

Set $T = \sum_{i=1}^{2n} s_i + \sum_{i=1}^{3} p_{2n+i} + \sum_{i=1}^{2n} p_{i2} = 14nB$. We are ready to show that Equal-Size Partition has a solution.

Suppose that $|I_1| = m$. Then $\sum_{i\in I_1} s_i = \sum_{i\in I_1}(a_i + B) = mB + \sum_{i\in I_1} a_i$. So the sum of processing times of jobs and the setup times of processing batches is $T + \sum_{i\in I_1} s_i = 14nB + mB + \sum_{i\in I_1} a_i$ which is less than or equal to $Y = 15nB + B$. Hence, we have

$$|I_1| = m \le n \quad \text{and} \quad \sum_{i\in I_1} a_i \le B. \tag{6}$$

Now, the processing completion time of family F_{2n+2} is

$$\begin{aligned} Y_1 &= p_{2n+1} + \sum_{i\in I_1}(s_i + p_{i1}) + \sum_{i\in I_2}(s_i + p_{i1} + p_{i2}) + p_{2n+2} \\ &= \sum_{i\in I_1\cup I_2} s_i + p_{2n+2} + \sum_{i\in I_2} p_{i2} \\ &= 2nB + 2B + 4nB + (2n - m)3B \\ &= (2 + 12n - 3m)B. \end{aligned}$$

But the sum of the delivery times of jobs processed before F_{n+2} is

$$\begin{aligned} Y_2 &= t_{2n+1} + \sum_{i\in I_1} t_i + \sum_{i\in I_2} 2t_i \\ &= t_{2n+1} + \sum_{i\in I_1\cup I_2} t_i + \sum_{i\in I_2} t_i \\ &= 7nB + 3B + \sum_{i\in I_2}(2B - a_i) \\ &= (3 + 11n - 2m)B - \sum_{i\in I_2} a_i. \end{aligned}$$

Since there is no idle time in the transportation, we have $Y_1 \le Y_2$, and so $(n - m - 1)B + \sum_{i\in I_2} a_i \le 0$. This implies

$$|I_1| = m \ge n \quad \text{and} \quad \sum_{i\in I_2} a_i \le B. \tag{7}$$

By noting that $\sum_{i\in I_1} a_i + \sum_{i\in I_2} a_i = 2B$, from (6) and (7), we conclude that

$$|I_1| = m = n \quad \text{and} \quad \sum_{i\in I_1} a_i = \sum_{i\in I_2} a_i = B.$$

Consequently, I_1 and I_2 define a solution of the instance of Equal-Size Partition. The result follows. □

4.2. A general dynamic programming algorithm

In this subsection, we establish a general dynamic programming algorithm for the scheduling problem $1 \to D, f|v = 1, s_i, c_i = 1, t_i|C_{\max}$. From Lemma 2.1, suppose that the jobs within each family are indexed in SPT rule.

For $f + 1$ integers $k_1, k_2, \ldots, k_f, m$ with $0 \le k_i \le n_i, 1 \le i \le f$, and $1 \le m \le f$, we consider the problem $1 \to D, f|v = 1, s_i, c_i = 1, t_i|C_{\max}$ on the set of jobs $\{J_{ij} : 1 \le i \le f, 1 \le j \le k_i\}$ under the restriction that the job in the last delivery batch belongs to family F_m $(1 \le m \le f)$. Such a problem is denoted by $\mathcal{P}(k_1, k_2, \ldots, k_f, m)$. The following notations are used in our discussion.

- $R(k_1, k_2, \ldots, k_f, m)$ is the optimal objective value of $\mathcal{P}(k_1, k_2, \ldots, k_f, m)$.
- $\Pi(k_1, k_2, \ldots, k_f, m)$ is the set of all optimal SPT schedules of $\mathcal{P}(k_1, k_2, \ldots, k_f, m)$ assuming $R(k_1, k_2, \ldots, k_f, m)$.
- $\tau(k_1, k_2, \ldots, k_f, m)$ is the set of m' such that there is a schedule in $\Pi(k_1, k_2, \ldots, k_f, m)$ such that the second last job is in $F_{m'}$, $1 \le m' \le f$.
- $Y(k_1, k_2, \ldots, k_f, m)$ is the minimum processing completion time of last job among all schedules in $\Pi(k_1, k_2, \ldots, k_f, m)$.

The dynamic programming recursion will calculates all values of

$$R(k_1, k_2, \ldots, k_f, m), \qquad \tau(k_1, k_2, \ldots, k_f, m) \quad \text{and} \quad Y(k_1, k_2, \ldots, k_f, m)$$

in this order.

The boundary conditions for the dynamic programming are

$$\tau(k_1, k_2, \ldots, k_f, m) = \begin{cases} \{m\}, & \text{if } k_i = 0 \text{ for all } i \ne m \text{ and } k_m = 1, \\ \emptyset, & \text{otherwise.} \end{cases}$$

$$Y(k_1, k_2, \ldots, k_f, m) = \begin{cases} s_m + p_{m1}, & \text{if } k_i = 0 \text{ for all } i \ne m \text{ and } k_m = 1, \\ +\infty, & \text{otherwise.} \end{cases}$$

$$R(k_1, k_2, \ldots, k_f, m) = \begin{cases} s_m + p_{m1} + t_m, & \text{if } k_i = 0 \text{ for all } i \neq m \text{ and } k_m = 1, \\ +\infty, & \text{otherwise.} \end{cases}$$

Write $\delta_{ij} = 0$ if $i = j$, and $\delta_{ij} = 1$ if $i \neq j$. Then the recursion for the dynamic programming can be given by

$$R(k_1, k_2, \ldots, k_f, m) = t_m + \min_{1 \leq m' \leq f} \max \begin{cases} R(k_1, \ldots, k_{m-1}, k_m - 1, k_{m+1}, \ldots, k_f, m'), \\ Y(k_1, \ldots, k_{m-1}, k_m - 1, k_{m+1}, \ldots, k_f, m') + \delta_{mm'} s_m + p_{k_m}. \end{cases}$$

$\tau(k_1, k_2, \ldots, k_f, m)$ is given by set of m' with $1 \leq m' \leq f$ such that

$$R(k_1, k_2, \ldots, k_f, m) = t_m + \max \begin{cases} R(k_1, \ldots, k_{m-1}, k_m - 1, k_{m+1}, \ldots, k_f, m'), \\ Y(k_1, \ldots, k_{m-1}, k_m - 1, k_{m+1}, \ldots, k_f, m') + \delta_{mm'} s_m + p_{k_m}. \end{cases}$$

$$Y(k_1, k_2, \ldots, k_f, m) = \min_{m' \in \tau(k_1, k_2, \ldots, k_f, m)} \{Y(k_1, \ldots, k_{m-1}, k_m - 1, k_{m+1}, \ldots, k_f, m') + \delta_{mm'} s_m + p_{k_m}\}.$$

The optimal value is given by $\min_{1 \leq m \leq f} \{R(n_1, n_2, \ldots, n_f, m)\}$.

The dynamic programming function has at most $f(n_1 + 1)(n_2 + 1) \cdots (n_f + 1) \leq f(n/f + 1)^f$ states. And each recursion runs in $O(f)$ time, since we have at most f choices for m' with $1 \leq m' \leq f$. Hence, the overall complexity of the above dynamic programming recursion is $O(f^2 n^f)$.

One interesting corollary of the above discussion is that, when $f = 1$, the problem becomes the proposed problem by Ahmadi et al. [1], which can be solved in $O(n)$ time (if the jobs are pre-indexed in SPT order).

4.3. A heuristic algorithm

In Section 3, we have shown that algorithm GT is optimal for $1 \to D, f|v = 1, s_i, c_i, t_i, GT|C_{\max}$. In this subsection, we will show that the objective value of schedule obtained by algorithm GT can be at most 3/2 times the optimal value of the problem $1 \to D, f|v = 1, s_i, c_i, t_i|C_{\max}$. We use σ and $C_{\max}(\sigma)$ to denote the schedule generated by algorithm GT and its corresponding objective value.

Let $C^*_{\max}$ be the optimal objective value of $1 \to D, f|v = 1, s_i, c_i, t_i|C_{\max}$. Note that for each family F_i, we define an associated 3-tuple (a_i, b_i, λ_i). The following lemma gives a lower bound for $C^*_{\max}$.

Lemma 4.2. $C^*_{\max} \geq \max\left\{\sum_{i=1}^f a_i, \sum_{i=1}^f b_i, \max_{1 \leq i \leq f}\{C_{\max}(\pi_i)\}\right\}$.

Proof. To justify the above inequality is a valid lower bound, we note that the first two terms in the maximization are the smallest total loads on the machine and the vehicle, while $C_{\max}(\pi_i)$ is the optimal objective value obtained by considering only the jobs in family F_i. □

Theorem 4.3. $C_{\max}(\sigma)/C^*_{\max} \leq 3/2$.

Proof. Without loss of generality, we assume that the processing sequence of families is $(F_1, F_2, \ldots, F_f)$ in σ. From the execution of algorithm GT, we can see that, in schedule σ, each family forms only one processing batch and the jobs in each family are consecutively processed on the machine and delivered by the vehicle in the same order. By the definition of (a_i, b_i, λ_i) and the execution of algorithm GT, the objective value of σ can be written as

$$C_{\max}(\sigma) = \max_{1 \leq k \leq f} \left\{\sum_{i=1}^{k} a_i - \lambda_k + \sum_{i=k}^{f} b_i\right\}. \tag{8}$$

Suppose that the maximum in (8) is attained at $k = c$. From (8), we have

$$C_{\max}(\sigma) = \sum_{i=1}^{c} a_i - \lambda_c + \sum_{i=c}^{f} b_i. \tag{9}$$

If $F_c \in A_1$, then $a_c \leq b_c$. By Lemma 4.2 and Eq. (1), we have $a_c - \lambda_c \leq \frac{1}{2}(a_c - \lambda_c + b_c - \lambda_c) \leq \frac{1}{2} C_{\max}(\pi_c) \leq \frac{1}{2} C^*_{\max}$. From the execution of Step 3.2 in algorithm GT, we have $a_i \leq b_i$ for $1 \leq i \leq c - 1$. By Lemma 4.2, we obtain

$$C_{\max}(\sigma) = \sum_{i=1}^{c} a_i - \lambda_c + \sum_{i=c}^{f} b_i \leq \sum_{i=1}^{f} b_i + a_c - \lambda_c \leq (3/2) C^*_{\max}.$$

If $F_c \in A_2$, then $a_c > b_c$. By Lemma 4.2 and Eq. (1), we have $b_c - \lambda_c < \frac{1}{2}(a_c - \lambda_c + b_c - \lambda_c) \leq \frac{1}{2} C_{\max}(\pi_c) \leq \frac{1}{2} C^*_{\max}$. From the execution of Step 3.2 in algorithm GT, we have $a_i > b_i$ for $c + 1 \leq i \leq f$. By Lemma 4.2, we obtain

$$C_{\max}(\sigma) = \sum_{i=1}^{c} a_i - \lambda_c + \sum_{i=c}^{f} b_i \leq \sum_{i=1}^{f} a_i + b_c - \lambda_c \leq (3/2) C^*_{\max}.$$

The result follows. □

References

[1] J.H. Ahmadi, R.H. Ahmadi, S. Dasu, C.S. Tang, Batching and scheduling jobs on batch and discrete processors, Operations Research 39 (1992) 750–763.
[2] A. Allahverdi, C.T. Ng, T.C.E. Cheng, M.Y. Kovalyov, A survey of scheduling problems with setup times or costs, European Journal of Operational Research 187 (2008) 985–1032.
[3] K.R. Baker, M.J. Magazine, Minimizing maximum lateness with job families, European Journal of Operational Research 127 (2000) 126–139.
[4] J. Bruno, P. Downey, Complexity of task sequencing with deadlines, setup times and changeover cost, SIAM Journal on Computing 7 (1978) 393–404.
[5] Y.C. Chang, C.Y. Lee, Machine scheduling with job delivery coordination, European Journal of Operational Research 158 (2004) 470–487.
[6] B. Chen, C.Y. Lee, Logistics scheduling with batching and transportation, European Journal of Operational Research 189 (2008) 871–876.
[7] S.S. Erengüc, N.C. Simpson, A.J. Vakharia, Integrated production/distribution planning in supply chains, European Journal of Operational Research 115 (1999) 219–236.
[8] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.
[9] N.G. Hall, C.N. Potts, Supply chain scheduling: Batch and delivery, Operational Research 51 (2003) 566–583.
[10] Y. Ikura, M. Gimple, Scheduling algorithms for a single batch processing machine, Operations Research Letters 5 (1986) 61–65.
[11] F. Jin, S.J. Song, C. Wu, A simulated annealing algorithm for single machine scheduling problems with family setups, Computers and Operations Research 36 (2009) 2133–2138.
[12] C.Y. Lee, Z.L. Chen, Machine scheduling with transportation considerations, Journal of Scheduling 4 (2001) 3–24.
[13] C.L. Li, G. Vairaktarakis, C.Y. Lee, Machine scheduling with deliveries to multiple customer locations, European Journal of Operational Research 164 (2005) 39–51.
[14] M.M. Liaee, H. Emmons, Scheduling families of jobs with setup times, International Journal of Production Economics 51 (1997) 165–176.
[15] M.M. Mazdeh, M. Sarhadi, K.S. Hindi, A branch-and-bound algorithm for single-machine scheduling with batch delivery minimizing flow times and delivery costs, European Journal of Operational Research 183 (2007) 74–86.
[16] C.N. Potts, M.Y. Kovalyov, Scheduling with batching: A review, European Journal of Operational Research 120 (2000) 228–249.
[17] C.N. Potts, L.N. Van Wassenhove, Integrating scheduling with batching and lot-sizing: A review of algorithms and complexity, Journal of the Operational Research Society 43 (1992) 395–406.
[18] J. Schaller, Scheduling on a single machine with family setups to minimize total tardiness, International Journal of Production Economics 105 (2007) 329–344.
[19] D.T. Thomas, P.M. Griffin, Coordinated supply chain management, European Journal of Operational Research 94 (1996) 1–15.
[20] S.T. Webster, K.R. Baker, Scheduling groups of jobs on a single machine, Operations Research 43 (1995) 692–703.
[21] W.Y. Zhong, G. Dösa, Z.Y. Tan, On the machine scheduling problem with job delivery coordination, European Journal of Operational Research 182 (2007) 1057–1072.